Doichain

(working draft)

The Atomic “Double-Opt-In“ and e-mail spam protection system on the blockchain

Version 0.0.5

5th of July 2018

Nico Krause
(nico@le-space.de)

www.doichain.org

Abstract:

We propose a decentralized peer-to-peer blockchain system that allows entities to securely request and store a proven permission to send an e-mail to another entity. We call this permission "atomic double-opt-in".

Contents:

4. **Mail Server provider and internet service provider who let customers freely use their mailbox.**
   Goal: Ensuring that valid e-mails are delivered into the inbox, while spam mails are sorted in the spam directory or blocked completely, all in the interest to satisfy their customers.

   Risk so far: Desired and awaited emails from customers are blocked and not delivered.

5. **Address generators**, **firms that specialise into campaigns like lotteries, in order to get agreements and addresses for their sponsors, who will use them in email marketing campaigns.**
   Goal: Generate the highest number of permissions possible with a relatively low amount of capital, for sponsors to use in their advertising campaigns.

6. **Courts**
   Goal: Definite prove, if a permission was granted for the delivery of this e-mail at an exact point in time.

3. Solutions with the aid of a Blockchain

1. The owner of an email address registers himself in an online-form and confirms his agreement.
2. The Address generators send the email addresses to a decentralized app (dApp) from the affected mail server provider. Next up the appropriate Single-Opt-In entry (SOI) takes place in the DOIChain.
3. The receiving node's DApp sends and email to the receiver and prompts him to confirm the asserted agreement with a click on a link.
4. The email receiver clicks on the DOI-link.
5. The DApp now confirms the Single-Opt-In (SOI) in the blockchain with a signature. As a result the SOI becomes a DOI, in other words it becomes a valid agreement, confirmed by the email address owner.

2. **Background and motivation**
   The initiators of Doichain are coming from a classical e-mail marketing background. As widely known, e-mail marketing industry struggles a lot with the quality of e-mail addresses and the whole process of sending e-mails in conjunction with spam. The main motivation was to create an atomic prove for high quality mailings by utilizing blockchain technology for requesting, confirming, storing and verifying an e-mail permission given by an entity and in a further step to help to identify spam and non-spam.

3. **Doichain dApp**
   The dApp works as an interface for humans and for machines to the blockchain.
   1. ***The machine2machine interface*** has a REST-API for three different modes:
      1. **Send-Mode:** Is used by parties which send, store SOIs, request unconfirmed and export confirmed DOIs. (we call this party Alice)
         1. **URL /opt-in (POST, authentication required) (Parameter: sender email, recipient email, auth token, userId)**
            Is usually called by the an application server of a website project which transfers the senders and recipients email address to the Doichain dApp. It stores the SOI (single-opt-in) on the Doichain and requests the DOI (double-opt-in) from the corresponding public key Doichain address, which was queried from the recipient emails DNS (TXT value).

            The DOI also contains a by the same public key encrypted URL of the local dApp (Alice). So the confirming dApp (Bob) can reqquest email template to be send to the final recipient (Peter). It is also used in order to inform the requesting (sending) dApp (Alice) about a successful DOI confirmation.

            The name_op "name_doi" transaction gets transferred to the recipient trusted dApp with an enabled "confirm-mode".

            Internally it creates a signature over a string containing the sender and recipients email address with the private key of the owner of the wallet.

         2. **/opt-in (PUT) (Parameter: nameId of DOI, signature of DOI)**
            Is called by the dApp confirm (Bob) when a DOI got confirmed successfully by a internet user (Peter) after receiving the doi request email. It's simply necessary to offchain-inform the requesting party (Alice) about a successful given double-opt-in. The confirmation is saved in the local database of the dApp of the doi-requesting party (Alice)

         3. **/doi-mail (GET) (Parameter: nameId of DOI, signature DOI)**
            Is called by the dApp in confirm mode (Bob) as it received the unconfirmed SOI transaction to be confirmed by the final recipient (Peter).

            This URL returns the template which will be afterwards sent over SMTP to Peter and a couple of other important data e.g. redirect-url's and email subject.

            In further improvement releases this URL might also return an unsigned raw transaction hash to be signed by Bob.

         4. **/export (GET, authentication required)**
            Is called by the application server of the DOI requesting party and simply

exports all succesfull confirmed DOI's in JSON format. In future this interface will undergo some significant extensions regarding to filtering possibilties.

2. ***Confirm-Mode: Is used by the party which receives unconfirmed SOI transactions (Bob/Peter) and transfers confirmed DOI's back to Alice***
    1. **/opt-in/confirm/:hash (GET)**
    Is usually called by the browser of the internet user owning the recipient address (Peter) after he received the doi-request email of the SMTP Server which has a connected Doichain dApp in confirm mode.
    The hash gets decoded to an ASCII database id, signes and stores the confirmation of the DOI in the local database as executes name_op "name_doi" on the Doichain blockchain so the DOI becomes officialy valid after 6 confirmations.

    2. **/walletNotify (GET) (Parameter: tx – the transaction id of incoming SOI)**
    Is called by a script of Bob's Doichain node as soon as a transaction arrives onto a public key (address) whichs private key he holds in his wallet.

    The transaction is checked if it contains a suitable name_op and name_doi. The name, value and address gets extracted from the Doichain and saved in the local database. Further on /doi-mail is called on the dApp which sent out the SOI in order to receive the email template for the DOI-Request email to be sent to Peter.

3. ***Verify-Mode: Can be used by all involved parties to verify and validate a given double-opt-in permission. In future in can be used by a SMTP filter to validate an incoming email which contains special DOI-smtp-headers as valid or invalid.***
    1. ***/verify (GET) (Parameter: name_id, signature, sender_email, recipient_email)***
    Reads the DOI-entry from the Doichain via the name_id (given in the SMTP-DOI-header). Validates two signatures:
        1. SOI - Signature stored in DOI-entry via public key given in the SMTP-DOI-Header over String build from sender_email and recipient_email
        2. DOI - Signature stored in DOI-entry via public key given by DNS TXT attribute over String build from the first signature.

2. ***The human interface*** with the following functions
    1. Account Balance (Doi)
    2. Query NameId's and DOI's
    3. Wallet Transactions
    4. Wallet Accounts & Adresses
    5. Walletfunctions (e.g. sendToAddress, MultiSig)
    6. Verify DOI's with sender and recipient address, public key and nameId
    7. Doi Requests and Confirmations (inkl. Recipient and Sender)

4. **Doichain Node**

Is a Namecoin (Bitcoin) based software fork with new genesis block and other modifications. The Dochain team adds a new rpc-command **“name_doi”** which takes two to three arguments:

name_doi "name" "value" ("toaddress")

name_doi registers the given name and value and transfers it to an optionally to a Doichain address:

Arguments:

1. "name"	(string, required) the name under which the doi will be registered. A name for the doichain project should always use the namespace “e/”followed by a 256-bit, ECDSA valid, number represanted as a 32 byte (64 characters) string (Same as every Bitcoin privateKey). See also: https://en.bitcoin.it/wiki/Private_key
2. "value"	(string, required) value for the name - containing all necessary information of a valid soi/doi
3. "toaddress"	(string, optional) address to send the doi to

Result:

"txid" (string) the name_doi's txid

Examples:

doichain-cli name_doi "myname", "new-value"

doichain-cli name_doi "myname", "new-value", "NEX4nME5p3iyNK3gFh4FUeU..."

5. **The atomic double-opt-in**
   A double-opt-in (DOI) is a two step procedure required by European Data Protection Law when a website onwer collects e-mail addresses from website users in order to ssend an e-mail to them. Typically a website interacts with an internet user and offers a subscription to an e-mail newsletter in one or the other way. The internet user has to actively opt-in his wish on this website to be legal.

   This opt-in is the first step and is also called the single-opt-in (SOI). As the user submits the newsletter subscription form, the application server of the website has to send out another e-mail, which the internet user has to confirm., typically by clicking a HTTP-link. The so created double-opt-in can be legally registered on the websites newsletter database.

   Interestingly, this process in practice is completely insecure and companies in this field are acting completely on trust when working with contractors or suppliers which acquire and sell DOI's.

   Further on, legally acquired DOI's tend to be often forgotten by the entities itself which once in fact gave them. In that case other annoyences are inevitable. All in all, the whole procedure is neither trustable, nor reliable or satisfying for all players involved so far.

   Doichain steps in here and for a first quick overview and easier understanding we assume a simple scenario:
   1. Sceneario with three players: Alice (website owner), Peter (internet user), Bob (mail provider)

   Alice owns a website with connected Doichain node and Doichain dApp. She wants to send a newsletter to Peter. We further assume the e-mail of Peter is hosted with Bob' mail-server with connected Doichain Node and dApp.

   1. At First, Peter submits his e-mail address on Alice's newsletter form while the application server of Alice saves Peters e-mail address locally and forwards the e-mail address of the sender (Alice's) and the recipient (Peter's) to her local Doichain dApp by posting to a REST-API.
   2. The Doichain dApp, generates unique Hash-key, which we call historically correct ***NameId*** (since Doichain uses mainly forked Namecoin functions). In Doichain we use the namespace “e/” for “e-mail” (instead of d/ or id/ in Namecoin). The NameId references SOI and later DOI on the blockchain with Peters e-mail address in the local database. In the current dApp prototype version, we generate and locally store a interims private- and public key pair for Peter, since we assume he doesn't have a Doichain wallet nor Doichain node yet.

      We create a signature over a text string using Peter's interim private key over the e-mail address of sender and recipient (here Alice's and Bob's).

      Now, we execute the ***name_doi*** Doichain RPC command on the blockchain which initially stores the single-opt-in (SOI) inside the blockchain. It will be able to to be found by the ***NameID*** and the signature can be verified by public key of Peter.

      Bob's mail-server has a connected Doichain node (and public-key), which we published via a ***DNS TXT record***. We send the name_op transaction NameID which points to the SOI to and address of this public key. As we

would send money on the Bitcoin network, the transaction gets broadcasted all over the connected Doichain nodes.

3. Bob's Doichain Node informs his dApp via walletnotfiy, which sends out an e-mail via SMTP into Peters e-mail inbox. The e-mail template to be used was gathered via REST from Alice dApp straight over HTTP(S)). Peter receives the DOI-request e-mail, confirms it with a HTTP-confirm link pointing to Bobs dApp. The dApp will executes a second ***name_doi*** command on the Doichain by signing the DOI with the private key of Bob's mail-server connected Doichain node.

4. Another day Alice decides to send out her newsletter and attaches two additional smtp header to her e-mail: X-Doichain_NameId and X-Doichain-Public-Key of Bob's DOI reference hash in the Doichain and Bob's public key.

5. Bob's SMTP Server has an installed smtp filter to lookup Doichain NameId's. It receives the e-mail from Alice and filters out the DOICHAIN_NAMEID and DOICHAIN_PUBLICKEY SMTP-headers and verifies it by contacting the local Doichain dApp via REST URL /verify. If the signature of the DOI is valid and corresponds with the senders and recipients e-mail address, the e-mail is eventually delivered to Bob's Inbox.

2. Sequence Diagram